Amendments to the Claims

The listing of claims will replace all prior versions, and listings of claims in the

application.

1.   (Currently Amended) An apparatus comprising:

a processor capable of simultaneous execution of two or more threads of instructions,

~~where said~~ wherein the processor comprises:

_____at least ~~one~~ two resource units, each capable of being assigned to an instruction

~~two or more~~ of each of the threads, wherein each of the resource units implements a portion of

instruction types occurring in each of the threads;

_____a priority register to store thread information for the threads, the thread

information including a priority code corresponding to the instructions of each thread, at least

one of the threads requesting use of one of the resource units for processing a current

instruction; and

_____a priority selector coupled to the priority register to generate an assignment signal

to assign the ~~at least one~~ resource units to the requesting thread's current instruction according to

the priority codes.

2.   (Currently Amended) The apparatus of claim 1 wherein each of the ~~at least one~~

resource units is one of an instruction fetch unit, a memory locking unit, a load unit, a store unit,

an input/output unit, a peripheral unit interface, and a functional unit.

3.   (Original) The apparatus of claim 2 wherein the functional unit is one of an

arithmetic unit, a logic unit, and an arithmetic and logic unit.

4.   (Original) The apparatus of claim 1 further comprising:

an instruction multiplexer coupled to the priority selector to pass instructions stored in a

plurality of instruction registers to execution units according to the assignment signal.

5.   (Currently Amended) The apparatus of claim 1 further comprising:

a priority assignor coupled to the priority register to set the thread information including

at least one of the priority codes corresponding to the at least one of instructions of the each

thread[[s]] in response to a start instruction from an instruction decoder and dispatcher.

6.   (Currently Amended) The apparatus of claim 5 wherein the priority assignor sets

an active flag in the priority register corresponding to the at least one of the threads in response

to the start instruction.

7.   (Currently Amended) The apparatus of claim 6 wherein the priority assignor

resets the active flag in the priority register corresponding to the at least one of the threads in

response to a quit instruction from the instruction decoder and dispatcher.

8.   (Currently Amended) The apparatus of claim 1 wherein the priority selector

assigns the at least one of the resource units to the instructions of at least one of the threads if the

at least one of the threads is not served and the at least one of the resource units is free.

9.   (Currently Amended) The apparatus of claim 8 wherein the ~~at least~~ one of the threads has highest priority code among a set of ready threads.

10.   (Currently Amended) The apparatus of claim 8 wherein the priority selector iteratively assigns resource units to <u>instructions of</u> threads in the set of ready threads according to the corresponding priority codes and resource availability until the set becomes empty.

11.   (Currently Amended) A method comprising:

executing two or more threads of instructions simultaneously in a processor;

storing thread information for the threads, the thread information including a priority code corresponding to <u>instructions of</u> each thread, at least one of the [[P]] threads requesting use of ~~at least~~ one <u>of at least two</u> resource unit<u>s</u> capable of being assigned to <u>an instruction</u> ~~two or more~~ <u>of each</u> of the threads <u>for processing a current instruction, wherein each of the resource units implements a portion of instruction types occurring in each of the threads</u>; and

generating an assignment signal to assign the ~~at least one~~ resource unit<u>s</u> to the requesting thread<u>'s current instruction</u> according to the priority codes.

12.   (Currently Amended) The method of claim 11 wherein <u>each of</u> the ~~at least one~~ resource unit<u>s</u> is one of an instruction fetch unit, a memory locking unit, a load unit, a store unit, an input/output unit, a peripheral unit interface, and a functional unit.

13.   (Original) The method of claim 12 wherein the functional unit is one of an arithmetic unit, a logic unit, and an arithmetic and logic unit.

14.   (Original) The method of claim 11 further comprising:

passing instructions stored in a plurality of instruction registers to execution units

according to the assignment signal.

15.   (Currently Amended) The method of claim 11 further comprising:

setting the thread information including at least one of the priority codes corresponding

to the at least one of instructions of the each thread[[s]] in response to a start instruction from an

instruction decoder and dispatcher.

16.   (Currently Amended) The method of claim 15 wherein setting the thread

information comprises setting an active flag in the priority register corresponding to the at least

one of the threads in response to the start instruction.

17.   (Currently Amended) The method of claim 16 wherein setting the thread

information comprises resetting the active flag in the priority register corresponding to the at

least one of the threads in response to a quit instruction from the instruction decoder and

dispatcher.

18.   (Currently Amended) The method of claim 11 wherein generating the assignment

signal comprises generating the assignment signal to assign the at least one of the resource units

to the instructions of at least one of the threads if the at least one of the threads is not served and

the at least one of the resource units is free.

- 6 -

Jack B. DENNIS
Appl. No. 09/715,778

19.   (Currently Amended) The method of claim 18 wherein the at least one of the threads has highest priority code among a set of ready threads.

20.   (Currently Amended) The method of claim 1 wherein generating the assignment signal comprises iteratively assigning resource units to instructions of threads in the set of ready threads according to the corresponding priority codes and resource availability until the set becomes empty.

21.   (Currently Amended) A processor capable of simultaneous execution of two or more threads of instructions, comprising:

at least one a two resource units to provide resources for use by the threads, each capable of being assigned to an instruction two or more of each of the threads, wherein each of the resource units implements a portion of instruction types occurring in each of the threads; and

a resource prioritizer coupled to each of the resource units to prioritize resource utilization, the resource prioritizer comprising:

_____a priority register to store thread information for the threads, the thread information including a priority code corresponding to the instructions of each thread, at least one of the threads requesting use of one of the resource units for processing a current instruction, and

_____a priority selector coupled to the priority register to generate an assignment signal to assign the at least one resource units to the requesting thread's current instruction according to the priority codes.

22.   (Currently Amended) The processor of claim 21 wherein each of the at least one resource units is one of an instruction fetch unit, a memory locking unit, a load unit, a store unit, an input/output unit, a peripheral unit interface, and a functional unit.

23.   (Original) The processor of claim 22 wherein the functional unit is one of an arithmetic unit, a logic unit, and an arithmetic and logic unit.

24.   (Original) The processor of claim 21 the resource prioritizer further comprising:

an instruction multiplexer coupled to the priority selector to pass instructions stored in a plurality of instruction registers to execution units according to the assignment signal.

25.   (Currently Amended) The processor of claim 21 wherein the resource prioritizer further comprising:

a priority assignor coupled to the priority register to set the thread information including at least one of the priority codes corresponding to the at least one of instructions of the each thread[[s]] in response to a start instruction from an instruction decoder and dispatcher.

26.   (Currently Amended) The processor of claim 25 wherein the priority assignor sets an active flag in the priority register corresponding to the at least one of the threads in response to the start instruction.

27.   (Currently Amended) The processor of claim 26 wherein <u>the priority assignor</u> resets the active flag in the priority register corresponding to ~~the~~ at least one of the threads in response to a quit instruction from the instruction decoder and dispatcher.

28.   (Currently Amended) The processor of claim 21 wherein the priority selector assigns ~~the at least~~ one <u>of the</u> resource unit<u>s</u> to the <u>instructions of</u> ~~at least~~ one of the threads if the ~~at least~~ one of the [[P]] threads is not served and the ~~at least~~ one <u>of the</u> resource unit<u>s</u> is free.

29.   (Currently Amended) The processor of claim 28 wherein the ~~at least~~ one of the threads has highest priority code among a set of ready threads.

30.   (Currently Amended) The processor of claim 28 wherein the priority selector iteratively assigns resource units to <u>instructions of</u> threads in the set of ready threads according to the corresponding priority codes and resource availability until the set becomes empty.